CircleMUD Socials

Jeremy Elson

December 6, 2001

Abstract

This document is a description of how to create new 'social' commands in CircleMUD. C programming experience generally not required at all for creating new CircleMUD socials. The intended audience for this document is all CircleMUD administrators. This document ties together closely with the document on the `act()` function.

1 Introduction

A general system exists to handle the 'social' commands – those which generally have no game-related purpose other than to convey emotions between players. Socials are also useful for creating advanced DikuMud adventures and quests through the use of speical procedures; for example, you could add a 'detonate' social which, within your quest, is handled by a special procedure to cause something to explode.

Socials are all handled through the generalized command `do_action`. The text file `lib/text/socials` contains information on the game's socials. New socials can be added by 1) adding the name and level of the social to the master command list in `interpreter.c` (giving `do_action` as the function pointer), and 2) adding information for the new social to the `lib/text/socials` file.

2 File Format

In Circle 3.0, socials in the file are specified by name (instead of by command number, as was true of the original DikuMud and versions of CircleMUD before 3.0.). In the standard CircleMUD distribution, the socials appear in alphabetical order in the file, but they can appear in any order and can be rearranged if necessary.

The file is formatted as follows:

```
<command name> <hide-flag> <minimum position of victim>
<messg to character if no argument>
<messg to others if no argument>
<messg to char if victim found>           <---If this field is empty,
<messg to others if victim found>          <-
<messg to victim>                            | then these fields must be
<messg to char if victim not found>          | skipped, and the action will
<messg to char if vict is char>              | ignore arguments.
<messg to others if vict is char>          <-

<command name> <hide-flag> <minimum position of victim>
.
.
$~
```

Each social must contain either 1) only the first two messages (if the social ignores arguments), or 2) all eight messages (if the social takes an argument). Each message must be contained in one line.

The command-name indicates which social is being specified. The hide-flag can be either 0 or 1; if 1, the social is hidden from OTHERS if they cannot see the character performing the social. The action is not hidden from the VICTIM, even if s/he cannot see the character performing the social, although in such cases the character's name will, of course, be replaced with "`someone`".

Where it makes sense to do so, text fields may be left empty. This is done by putting a '#' in the first column on the line. Doing so is legal in the following fields:

- a: messg to others if no arg
- b: messg to others if victim found
- c: messg to others if vict is char

Note again that if the field *messg to char if victim found* is empty, then the following fields must be omitted entirely (not even the '~'), and the action will ignore any arguments supplied.

The procedure sends the text strings through `act()`, and they may contain control codes (see the documentation for the `act()` function for details.) Note that not all of `act()`'s control codes may be used; for example, codes which refer to objects such as `$p`.

For the sake of efficiency, no tests or sanity checks are made on the consistency or logic of the `act()` codes found in the social messages. Hence, grave disorder may result if, say, the code '`$N`' occurs in a text field that does not refer to a victim; like *messg to others if no arg*. In previous versions of Circle, such grave disorder often manifested itself in the form of crashes. However, the `act()` function itself in CircleMUD 3.0 and above does sanity checks to make sure it is not dereferencing null pointers, hopefully with the result of avoiding crashes. Nevertheless, great care should be taken in making sure that the $ codes of socials are used consistently and correctly.

More information can be found in the documentation on the `act()` function.